Rapport Projet Machine Learning: Octroie de Crédit

Laakad Mouad

I- Présentation de la Problématique :

Lending Club est une société de prêt entre pairs basée aux États-Unis, dans laquelle les investisseurs fournissent des fonds aux emprunteurs potentiels et les investisseurs réalisent un profit en fonction du risque qu'ils prennent. Lending Club fournit le "pont" entre les investisseurs et les emprunteurs. Pour plus d'informations de base sur l'entreprise, veuillez consulter l'article de wikipedia sur l'entreprise : https://en.wikipedia.org/wiki/LendingClub

L'objectif du projet est de construire une application qui permet de prédire Solvabilité des clients potentiels et ainsi décider Octroi des crédits et ceci en suivant les étapes suivantes :

- EDA : Analyse exploratoires des données d'anciens Clients

- Data Preprocessing : Prétraitement des données pour préparation au modèle Machine Leanrning

- Application de 10 Modèles de classification sur notre data

- choix de 3 Modèles les plus Performants

- Choix du Modèle Finale

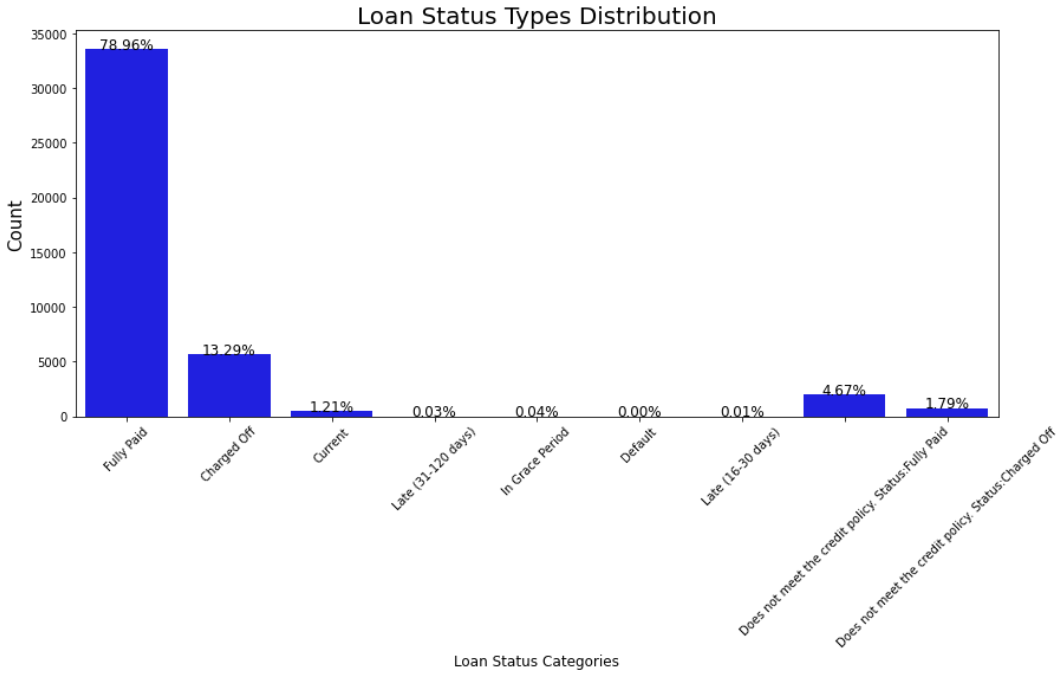
- Déploiement du Modèle via Streamlit en Local

- Publication de l'application sur le Cloud (Heroku)

Nb : le détail du code est présenté dans le Notebook : Laakad_Mouadlending_club_Project.ipynb

On présentera par la suite dans ce rapport une synthèse du travail réalisé dans ce projet .

II- EDA : Exploratory Data Analysis



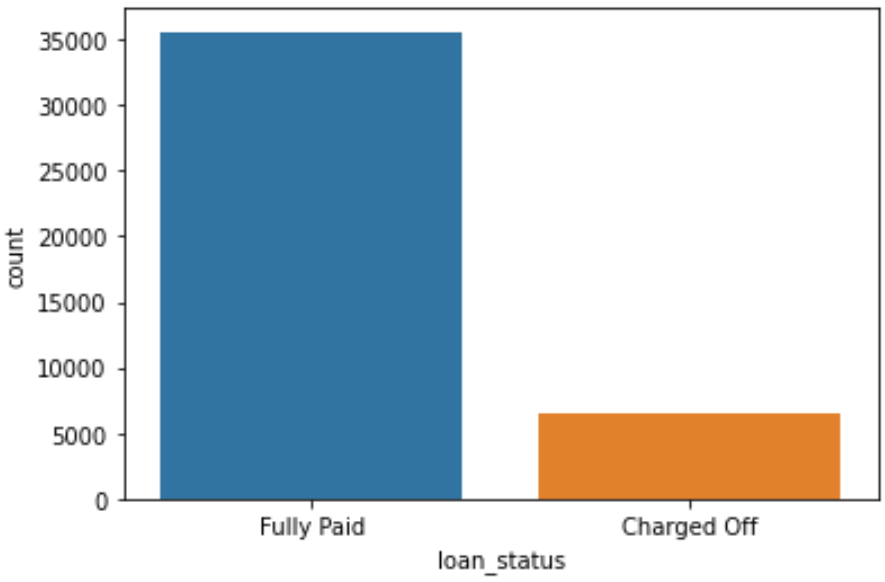
Pour notre variable Target , Notre intérêt portera surtout sur les valeurs : Fully Paid , Charged-off .

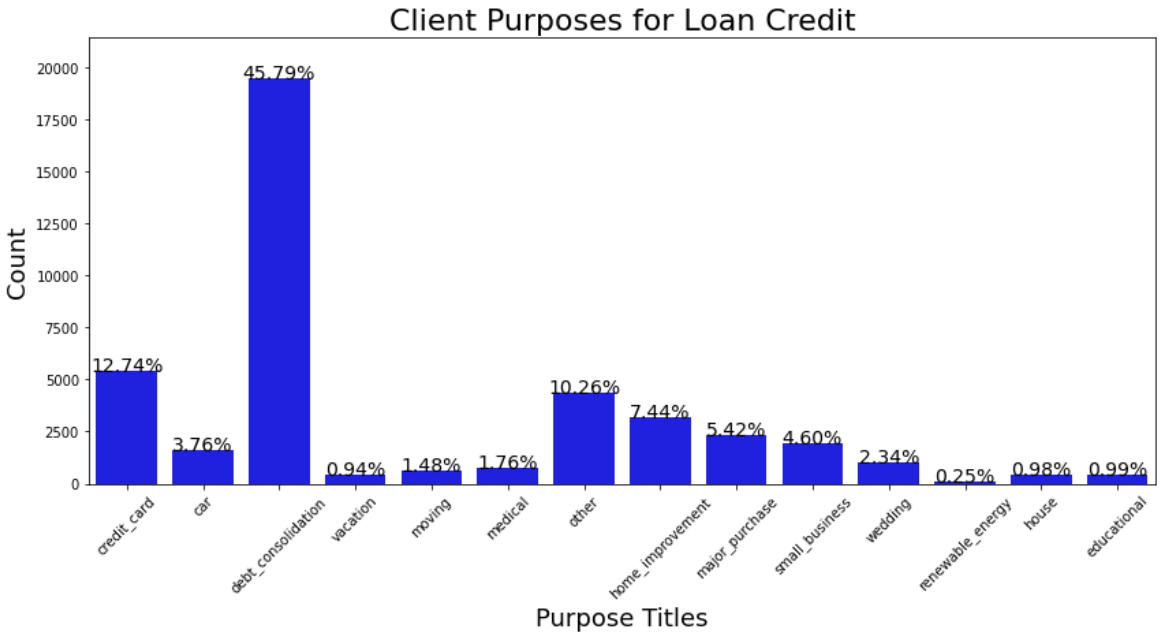
Notre étude est un sujet de classification pour prédire si un crédit sera payé(fully paid) or perdu(charged off) on procéde à la suppression des données qui concerne des crédit en cours (current)&(In grace periode)(on est pas certain s'il sera payé dans le futur)

ce qui sont en retard du crédit a partir de 15 jours sont susceptibles de ne pas payer le crédit , car la période de grâce (ne pas payer le montant mensuel pour une durée ne doit pas dépasser 15 jours) source : https://www.crowdfundinsider.com/2017/02/96585-lending-club-alters-grace-period-borrowers/

On supprime ainsi les données qui concernent (current)&(In grace periode) et on considère les catégories de retard comme des charged off

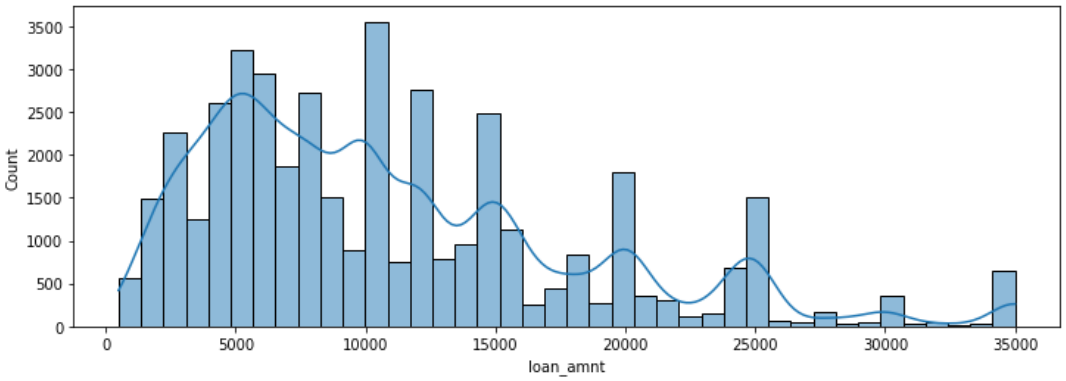
Résultat :

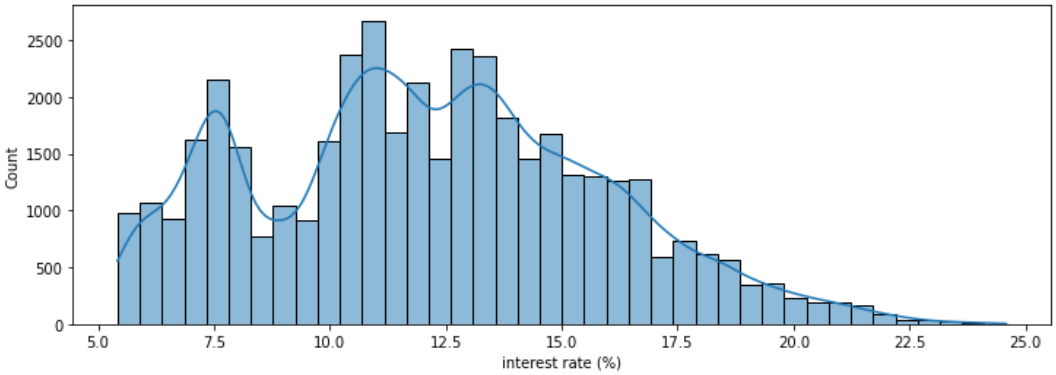




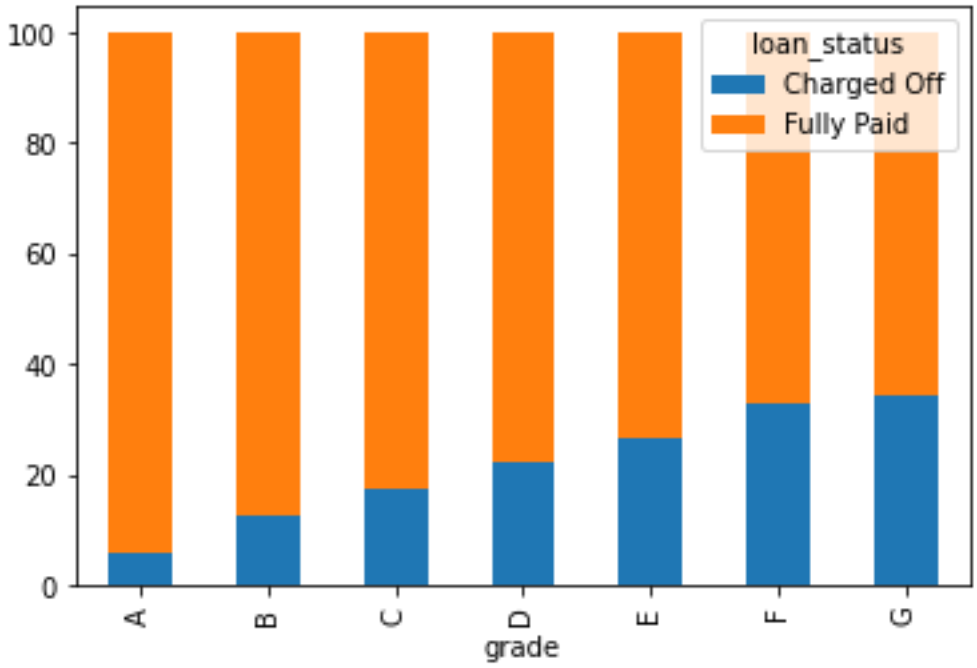
Les Top 3 purposes are:

1) 45.5% sont dédiés au :Debt Consolidation
2) 12.74% pour payer les Credit Card
3) 7.44% pour Home Improvement
4) and les autres sont 34.32%





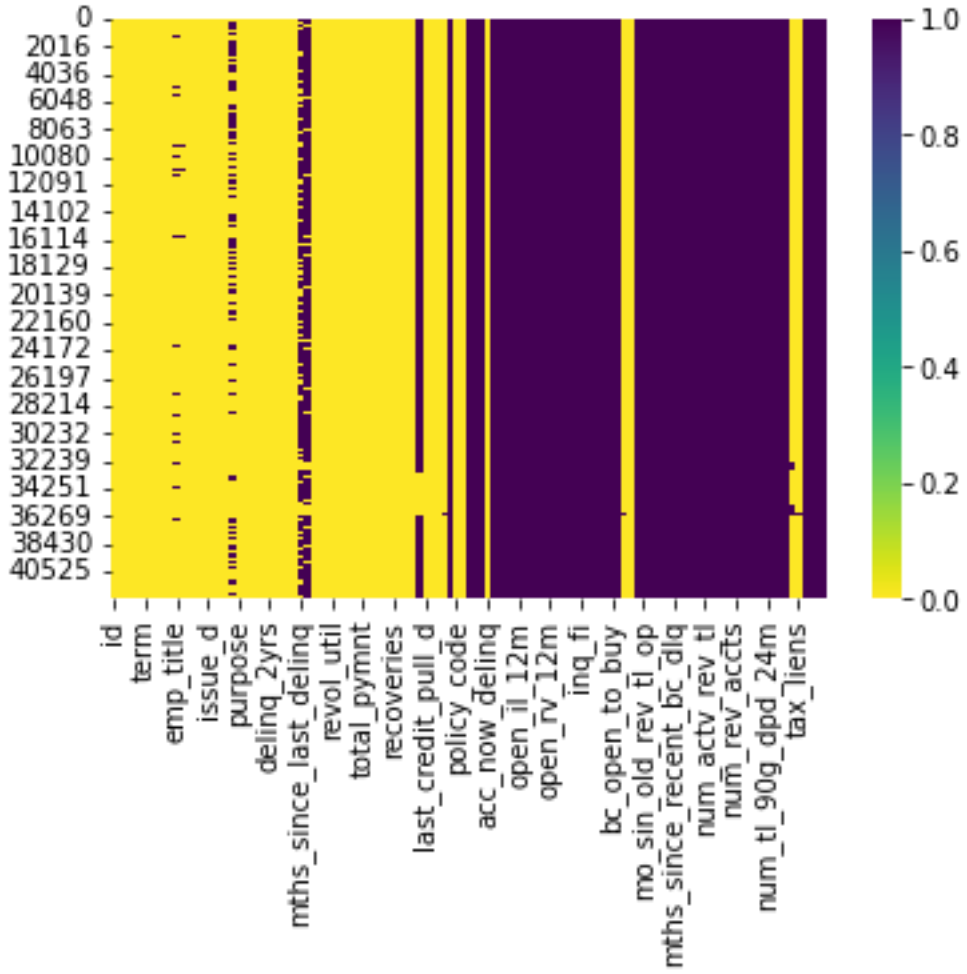
- une grande partie des prêts sont des valeurs jusqu a 10.000 USD
- une grande partie des taux d'intéret prêts et entre 7% et 15 %



On remarque plus on diminue dans le grade, plus le risque de non solvabilité augmente

III- *Data Preprocessing : Prétraitement des données pour préparation au modèle Machine Leanrning :*

1- DATA Cleaning



Ce graphe représente les missing values de notre data set :

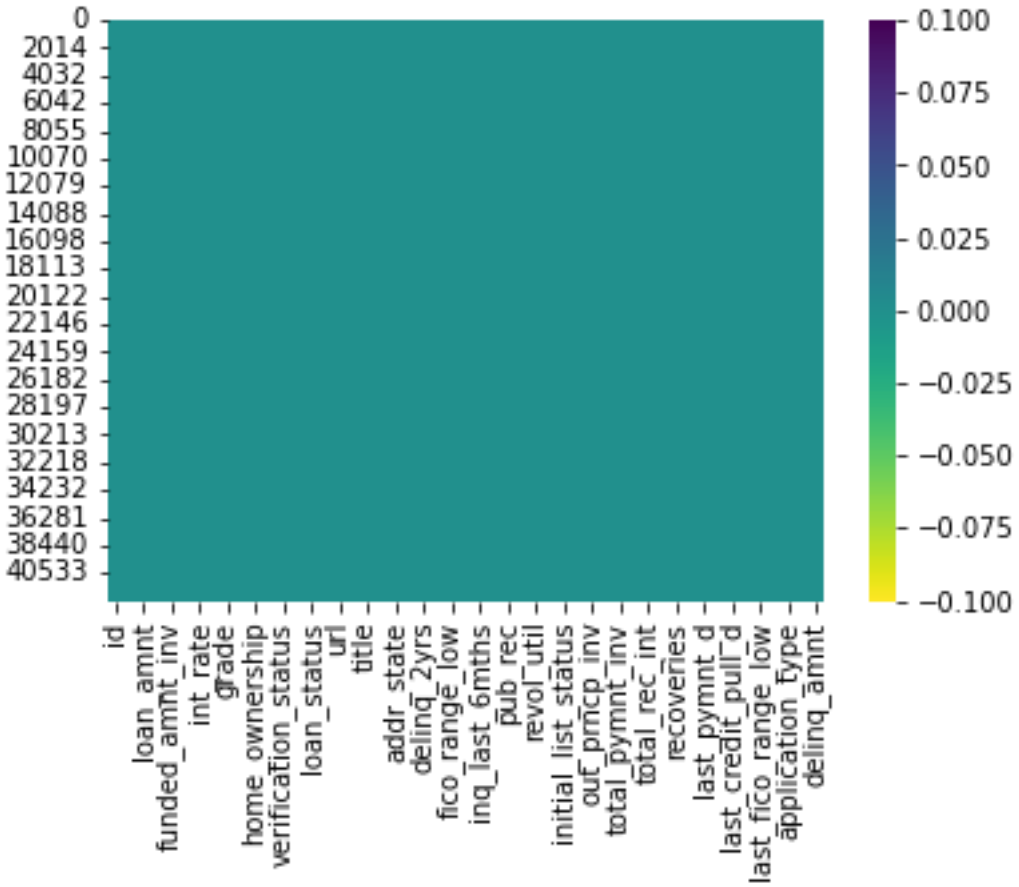
En Mauve : Missing values

En jaune : présence de données :

Opérations de cleaning réalisées :

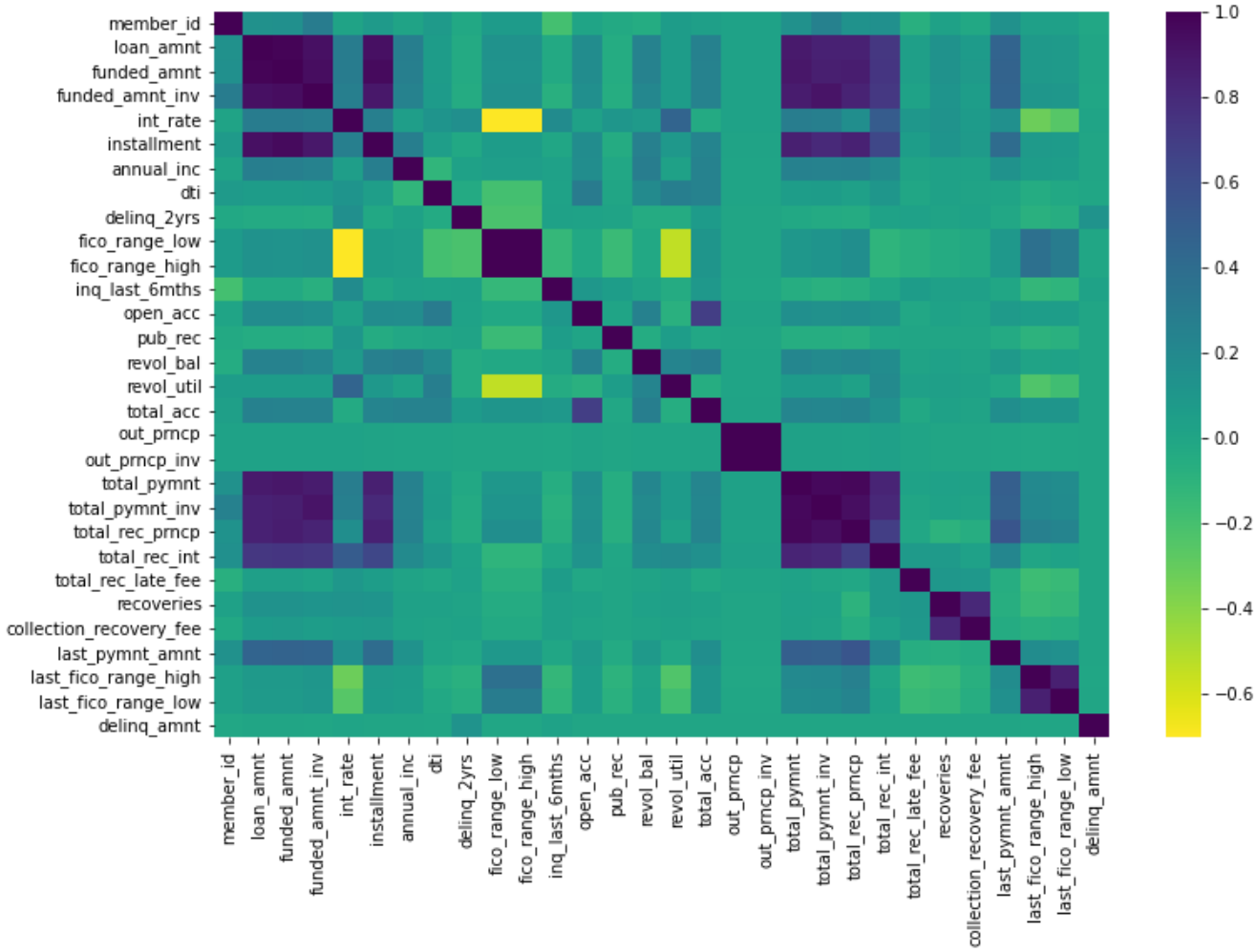
5) Suppression des colonnes ayant 100% de données NAN
6) Certaines colonnes ayant des valeurs Null contiennent une seul catégorie de données (False , individual ,) n'ajouteront pas grande performance pour notre future modèle
7) Il nous restent ainsi 32 lignes ayant des Null , leur suppression n'impactera pas la taille de notre data set (42000)

Résultat :



Aucune Valeur Null présente dans notre DATASET.

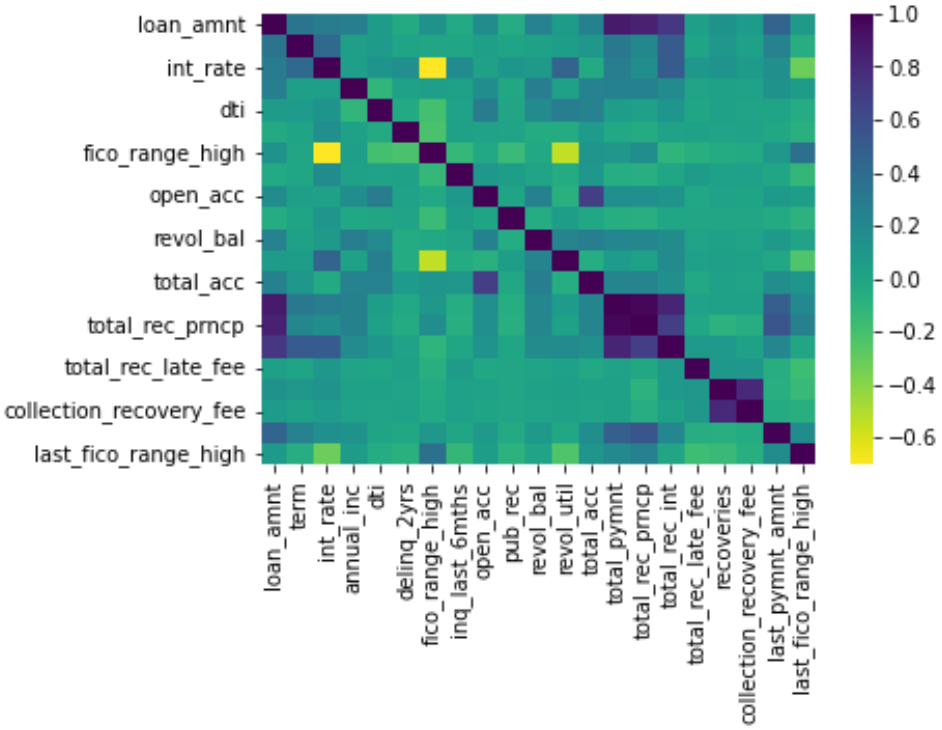
2. Data Preprocessing :

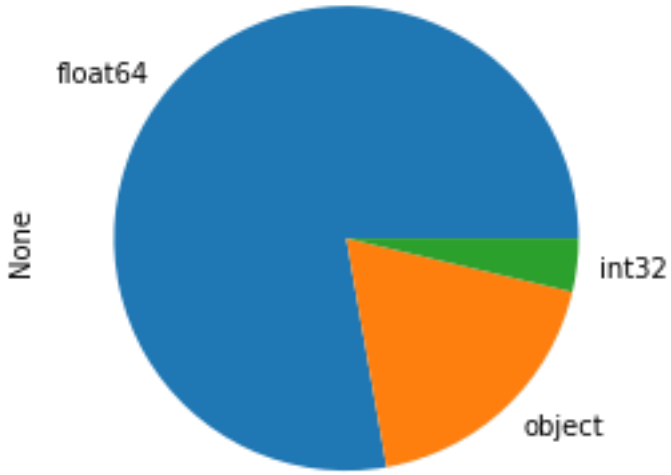


8) Ce graphe montre le taux de corrélation entre les variables de notre DATASET :
9) On sera amené à choisir notre colonnes de telle façon on réduit l'impact de la corrélation sur notre modèle.
10) Les 2 colonnes id et url fournissent la même information, L id est la fin de de L'URL
11) De même la colonne id ne fournira aucune info pour nos futurs modèles
12) fico_range_high et 'fico_range_low' 100% corrélés , on supprime la colonne 'fico_range_low' .
13) On a les données address state and zip code on supprime ces colonnes concernent la position géographique de la personne et n'influence pas la solvabilité du client

14) la colonne "verification_status" contient 'Source Verified' et 'Verified' qui signifient la meme chose , on remplace 'Source Verified' par 'Verified'

Résultat :





Ainsi pour la préparation du modèle, on aura affaire a encoder les variables catégoriques en Numérique

3- Encoding et Normalisation :

On utilisera le LabelEncoder comme outil d'encodage pour les variables catégoriques

Et on enregistra pour chaque colonne son encoder : sous format pkl ex : encoder_sub_grade.pkl

	sub_grade	home_ownership	verification_status	loan_status	purpose
0	B2	RENT	Verified	Fully Paid	credit_card
1	C4	RENT	Verified	Charged Off	car
2	C3	OWN	Not Verified	Fully Paid	debt_consolidation
3	B1	RENT	Not Verified	Charged Off	debt_consolidation
4	C2	RENT	Verified	Fully Paid	debt_consolidation
...	...	...	...	...	...
42533	C2	MORTGAGE	Verified	Fully Paid	wedding
42534	C2	RENT	Verified	Fully Paid	debt_consolidation
42535	D1	RENT	Not Verified	Fully Paid	other
42536	E2	OWN	Verified	Fully Paid	other
42537	B4	RENT	Verified	Charged Off	debt_consolidation

Résultat :

	sub_grade	home_ownership	verification_status	loan_status	purpose
0	6	4	1	1	1
1	13	4	1	0	0
2	12	3	0	1	2
3	5	4	0	0	2
4	11	4	1	1	2
...	...	...	...	...	...
42533	11	0	1	1	13
42534	11	4	1	1	2
42535	15	4	0	1	9
42536	21	3	1	1	9
42537	8	4	1	0	2

41961 rows × 5 columns

On regroupe après les données catégoriques encodés and les données numériques :

```
data3=pd.concat([cat_data,num_data],axis=1)
```

```
data3
```

	sub_grade	home_ownership	verification_status	loan_status	purpose	loan_amnt	term	int_rate	installment	annual_inc	dti	delinq_2yrs	fico_range_
0	6	4	1	1	1	5000.0	36.0	0.1065	162.87	24000.0	27.65	0.0	
1	13	4	1	0	0	2500.0	60.0	0.1527	59.83	30000.0	1.00	0.0	
2	12	3	0	1	2	6500.0	60.0	0.1465	153.45	72000.0	16.12	0.0	
3	5	4	0	0	2	6200.0	36.0	0.0991	199.80	25000.0	20.64	0.0	
4	11	4	1	1	2	14000.0	36.0	0.1427	480.33	35000.0	8.40	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
42533	11	0	1	1	13	5000.0	36.0	0.1349	169.66	113000.0	12.96	0.0	
42534	11	4	1	1	2	14975.0	60.0	0.1349	344.50	49100.0	15.71	0.0	

IV- Application de 10 Modèles de classification sur notre data :

Etapes :

15) Subdivision de données en données de training et données de test
16) Le scaling a été entrainé sur les données train et a permis de transformer les 2 données : X_train et X_test pour objectif d'éviter le data leakage entre le train set et le test set
17) Choix de 10 Modèles de classification
18) Evaluation des modèle a l'aide du F1 Score

Le choix du F1 comme indicateur de performance du modèle a été fait suivant :

- Le F1-Score combine subtilement la précision et le recall . Il est intéressant et plus intéressant que l'accuracy
- Dans les situations d'imbalanced class comme notre cas 85% sont Fully Paid et Charged off 15% , nous avons une majorité de vrais positifs qui faussent complètement notre perception de la performance de l'algorithme

$$\text{F1-Score} = 2\,\frac{\text{precision} * \text{recall}}{(\text{precision} + \text{recall})}$$

Résultat de l'évaluation :

```
1-LogisticRegression :
La precision du Modèle est :0.9806
------------------------------
2-KNeighborsClassifier :
La precision du Modèle est :0.9518
------------------------------
3-DecisionTreeClassifier :
La precision du Modèle est :0.9773
------------------------------
4-Support Vector Machine :
La precision du Modèle est :0.9834
------------------------------
5-Naive Bayes (Gaussian) :
La precision du Modèle est :0.975
------------------------------
6-Naive Bayes (Multinomial) :
La precision du Modèle est :0.9203
------------------------------
7-Stochastic Gradient Descent Classifier :
La precision du Modèle est :0.9779
------------------------------
8-Random Forest :
La precision du Modèle est :0.9957
------------------------------
9-Gradient Boosting Classifier :
La precision du Modèle est :0.9959
------------------------------
10-Adaptive Boosting Classifier  :
La precision du Modèle est :0.9961
------------------------------
```

D'aprés le choix du F1 Score : les Top 3 modèles sont :

- Random Forest : 99,57 %

- Gradient Boosting Classifier : 99,59 %

-Adaptive Boosting Classifier : 98,61 %¶

IV- choix de 3 Modèles les plus Performants

On évaluera chaque modèle des 3 modèle choisies suivant 3 données :

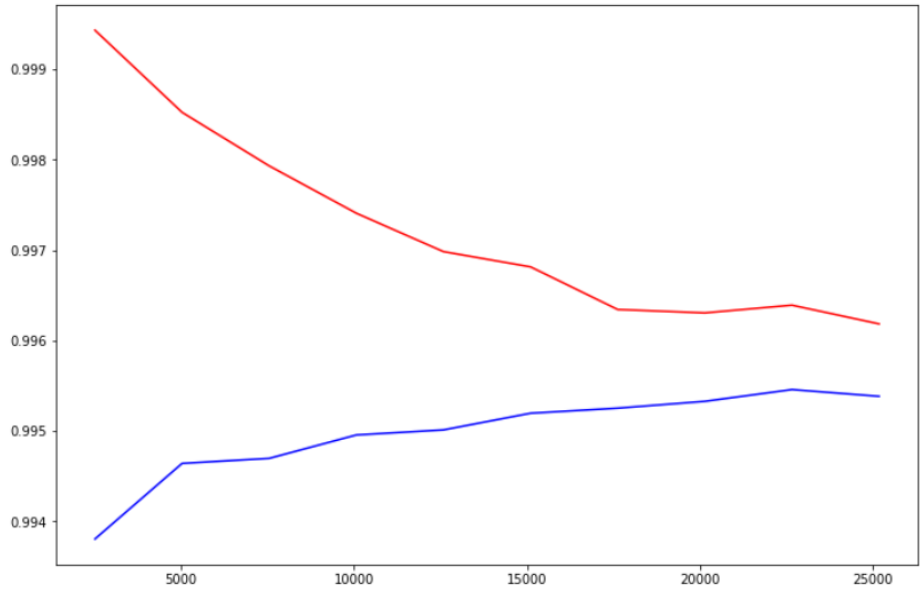
- Données Non balancés
- Données Balancés avec Oversampling : on équilibre notre dataset en augmentant le nombre de crédits charged- off
- Données Balancés avec Undersampling : on équilibre notre dataset en diminuant le nombre de crédits Fully paid

Ex : Oversampling :

```
Before OverSampling, counts of label '1': 26652
Before OverSampling, counts of label '0': 4818

After OverSampling, counts of label '1': 26652
After OverSampling, counts of label '0': 26652
```

Graphe d'évaluation :

```
: evaluation(model , X_train , y_train)
[[1543   63]
 [  10 8875]]
              precision    recall  f1-score   support

           0       0.99      0.96      0.98      1606
           1       0.99      1.00      1.00      8885

    accuracy                           0.99     10491
   macro avg       0.99      0.98      0.99     10491
weighted avg       0.99      0.99      0.99     10491
```



On aura ainsi 9 graphes d' évaluation et on choisira pour chaque modèle des 3 modèles celui après être entrainé sur un type de données donne un meilleure résultat

V- Choix du modèle finale et tuning

1- Choix du Modèle Finale

```
rée [139]: results
```

Out[139]:

	Accuracy	f1 Score	ROC AUC
Gradient Boosting Classifier	99.64	99.79	99.05
Random Forest	99.59	99.76	98.92
Adaptive Boosting Classifier	99.47	99.69	98.84

Conclusion :

Le Meilleure Modèle est le : Gradient Boosting Classifier

2- Tuning du Modèle Final

```
Entrée [150]: y_pred=Model1_tuned.predict(X_test)
              print(confusion_matrix(y_test,y_pred))

[[1454  152]
 [ 300 8585]]

Entrée [152]: y_pred=Model1.predict(X_test)
              print(confusion_matrix(y_test,y_pred))

[[1577   29]
 [   9 8876]]
```

On réalise du tuning sur notre modèle Gradient Boosting Classifier a l'aide de la librairie : **GridSearchCV**

On remarque que notre modèle avant modèle est plus performant

Finalement on enregistre le modèle sous format Pkl à l'aide de la librairie pickle

VI- Déploiement du Modèle via Streamlit en Local

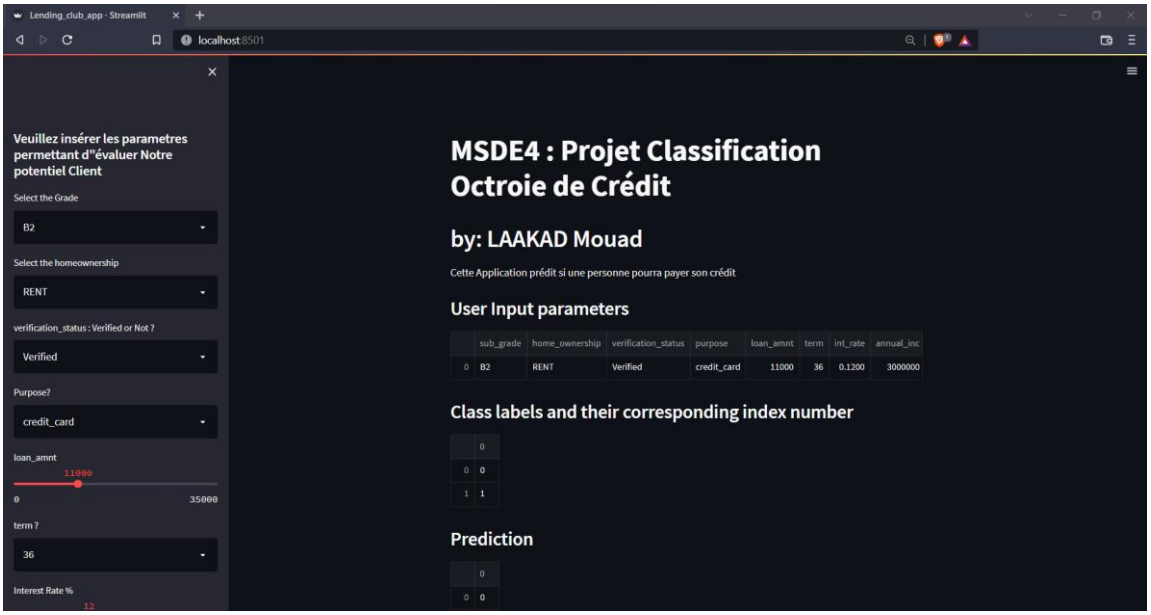
On réalise notre application qui permet de :

- saisir les données utilisateur à partir de l'interface web
- encoder via les modèle d'encodage des données catégoriques
- Réaliser le scaling des données
- Applique le modèle et afficher le résultat de prédiction

code de Lending_club_app.py :

```python
df = user_input_features()
#######------------------Transformation de données pour Model-----------------#######
#-------------------------------------------------------------------------------------

le_subgrade=pickle.load(open("encoder_sub_grade.pkl",'rb'))
le_home_ownership=pickle.load(open("encoder_home_ownership.pkl",'rb'))
le_verification_stat=pickle.load(open("encoder_verification_status.pkl",'rb'))
le_purpose=pickle.load(open("encoder_purpose.pkl",'rb'))
scaler=pickle.load(open("scaler.pkl",'rb'))
# fonction qui convertit une format yyyy-mm en timestamp

def transform_for_model(df):
    cat_data=df[["sub_grade","home_ownership" ,"verification_status","purpose"]]
    num_data=df[['loan_amnt', 'term', 'int_rate', 'annual_inc', 'dti',
       'delinq_2yrs', 'fico_range_high', 'inq_last_6mths', 'open_acc',
       'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'total_pymnt', 'total_rec_prncp', 'total_rec_int', 'total_rec_late_fee',
       'recoveries', 'collection_recovery_fee', 'last_pymnt_amnt',
       'last_fico_range_high']]
    cat_data["sub_grade"]=le_subgrade.transform(cat_data["sub_grade"])
    cat_data["home_ownership"]=le_home_ownership.transform(cat_data["home_ownership"])
    cat_data["verification_status"]=le_verification_stat.transform(cat_data["verification_status"])
    cat_data["purpose"]= le_purpose.transform(cat_data["purpose"])

    df=pd.concat([cat_data,num_data],axis=1)
    df=scaler.transform(df)
    return df
#######------------------Prédiction et affichage du résultat -----------------#######
#-------------------------------------------------------------------------------------
model=pickle.load(open("best_model.pkl",'rb'))

st.subheader('User Input parameters')
st.write(df)
df=transform_for_model(df)
if st.sidebar.button('Predict Solvability'):
  prediction = model.predict(df)
  prediction_proba = model.predict_proba(df)
  st.subheader('Class labels and their corresponding index number')
```

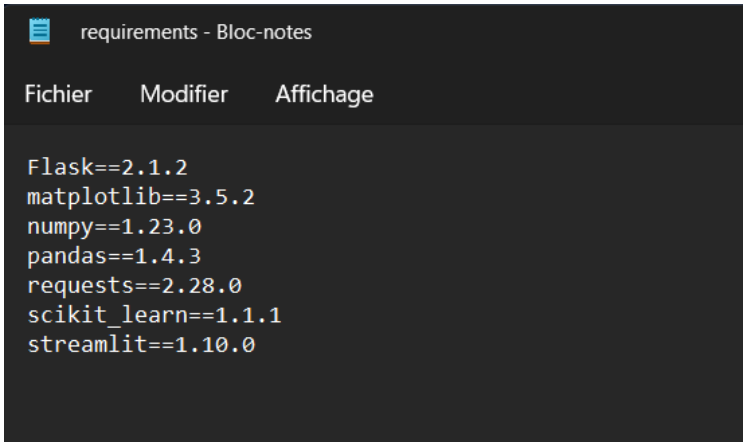
Commande sur terminal : Streamlit run Lending_club_app.py



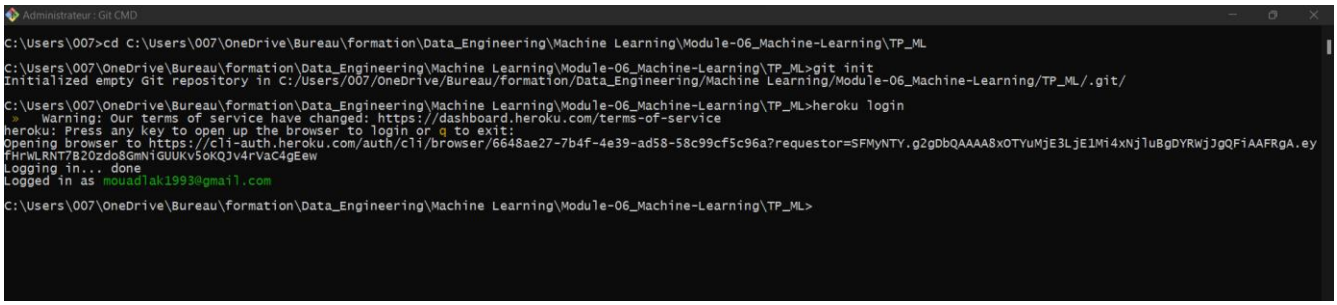
VII-Publication de l'application sur le Cloud (Heroku)

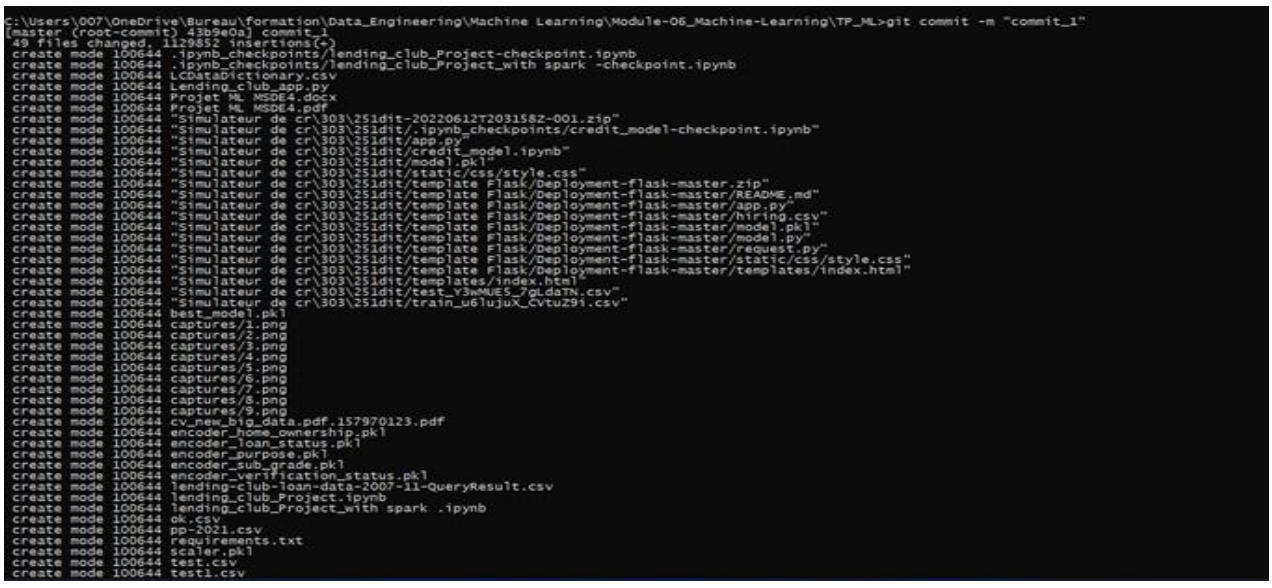
1) Création du fichier requirements.txt :

```
(my_env) PS C:\Users\007\OneDrive\Bureau\formation\Data_Engineering\Machine Learning\Module-06_Machine-Learning\TP_ML> p
ipreqs ./
INFO: Successfully saved requirements file in ./requirements.txt
(my_env) PS C:\Users\007\OneDrive\Bureau\formation\Data_Engineering\Machine Learning\Module-06_Machine-Learning\TP_ML>
```



2) Usage du GIT (version control pour ingester le programme sur heroku :





3) Envoyer les fichiers du programme vers heroku

```
C:\Users\007\OneDrive\Bureau\formation\Data_Engineering\Machine Learning\Module-06_Machine-Learning\TP_ML>git push heroku master
Enumerating objects: 56, done.
Counting objects: 100% (56/56), done.
Delta compression using up to 8 threads
Compressing objects: 100% (53/53), done.
Writing objects: 100% (56/56), 44.34 MiB | 891.00 KiB/s, done.
Total 56 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Determining which buildpack to use for this app
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the buildpack default: python-3.10.5
remote:        To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
remote: -----> Installing python-3.10.5
remote: -----> Installing pip 22.1.2, setuptools 60.10.0 and wheel 0.37.1
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote:        Collecting Flask==2.1.2
remote:          Downloading Flask-2.1.2-py3-none-any.whl (95 kB)
remote:        Collecting matplotlib==3.5.2
remote:          Downloading matplotlib-3.5.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.9 MB)
remote:        Collecting numpy==1.23.0
remote:          Downloading numpy-1.23.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.0 MB)
remote:        Collecting pandas==1.4.3
remote:          Downloading pandas-1.4.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.6 MB)
remote:        Collecting requests==2.28.0
remote:          Downloading requests-2.28.0-py3-none-any.whl (62 kB)
remote:        Collecting scikit_learn==1.1.1
remote:          Downloading scikit_learn-1.1.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (30.4 MB)
remote:        Collecting streamlit==1.10.0
remote:          Downloading streamlit-1.10.0-py2.py3-none-any.whl (9.1 MB)
remote:        Collecting itsdangerous>=2.0
remote:          Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
```

Ceci installe sur heroku les librairies précisés sur requirements.txt

4) Ouvrir note Application web de prédiction :

Lien : https://lending-club-project-2022.herokuapp.com